

APPLICATION  
FOR  
UNITED STATES LETTERS PATENT

TITLE: IMPROVING MULTICAST AND BROADCAST  
OPERATIONS IN ETHERNET SWITCHES

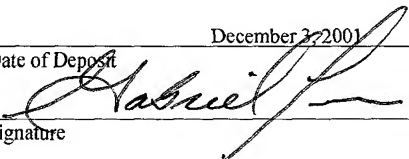
APPLICANT: RAHUL SAXENA

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL858858050US

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the U.S. Patent and Trademark Office, P.O. Box 2327, Arlington, VA 22202.

Date of Deposit December 3, 2001

Signature 

Gabe Lewis  
Typed or Printed Name of Person Signing Certificate

**IMPROVING MULTICAST AND BROADCAST**  
**OPERATIONS IN ETHERNET SWITCHES**

5

**TECHNICAL FIELD**

This invention relates to forwarding data frames through a switch, and more particularly to routing data frames for efficient multicast and broadcast operations in Ethernet switches.

10

**BACKGROUND**

15

20

Some switches, such as Ethernet switches, receive data frames at one or more ports. A data frame is an organized format of control or header data and payload data. The header data typically include fields such as the source address of the device transmitting the data frame, the destination address or addresses to which the data frame is being transmitted, length/type data indicating the length of the data frame as well as the data type of the payload data, and a frame check sequence field used as a form of error checking in verifying receipt of the data frame by the destination device. The control data are overhead that are used to assure that the payload data arrive at the destination device.

25

The payload data are the data of interest that are sent to the destination device. Examples of payload data include

pixel data for image rendering, audio data, text data and control data (e.g., commands requesting that the destination device transmit information back to the original source device).

5        In some network implementations, data frames may have different sizes. For example, in a typical Ethernet network, frame size may vary from a minimum of 64 bytes to a maximum of 1,518 bytes. In general, forwarding received data frames from a switch includes one of three methods: unicast, multicast and  
10 broadcast. A unicast transmission employs a one-to-one correspondence between the source device and the destination device such that the data frame output by a source device will only be received and used by a single destination device. A  
15 multicast transmission employs a one-to-many correspondence such that a single source device transmits a data frame for receipt and use by multiple destination devices. A broadcast transmission employs a one-to-all correspondence such that a single source device transmits a data frame to all destination devices on a particular network to which the source device is  
20 connected.

A switch receives data frames and forwards them to output ports for retransmission to other switches or destination devices. In some switches, a memory is employed to temporarily store a received data frame until the needed port

becomes free to output that data frame. These types of switches may be referred to as store-and-forward (SAF) switches.

Memory bandwidth usage can be inefficient as some of the  
5 memory bandwidth is not used when storing smaller data frames. To compensate for this, a memory is divided into independently addressable channels. This allows for smaller data frames to be stored in particular channels, which results in more efficient use of memory and increased throughput. That is,  
10 different channels may be accessed independently for reading or writing data frames. By dividing a large memory into channels, the overall speed, or useful bandwidth (data frame width multiplied by the memory access frequency), of a switch increases. By increasing the useful bandwidth of the internal  
15 hardware, additional ports may be supported without unnecessarily increasing the size of the memory.

When using an SAF switch to multicast or broadcast data frames over the network, a particular data frame may be read from memory multiple times, such as once for each port that is  
20 to retransmit the data frame. Since only one of the multiple channels is accessed at a time, the useful bandwidth of an SAF switch decreases when multicasting or broadcasting data frames.

## DESCRIPTION OF DRAWINGS

Fig. 1 is a block diagram of a switch.

Fig. 2 is a block diagram of a port of the switch of Fig.

1.

5 Fig. 3 is a block diagram of a copier, channel select logic circuit of the switch of Fig. 1.

Fig. 4a is a block diagram of a memory circuit in the copier, channel select logic circuit of Fig. 3.

10 Fig. 4b is a block diagram of a cell of the memory circuit of Fig. 4a.

Fig. 5 is a block diagram of a main memory circuit of the switch of Fig. 1.

Fig. 6 is a block diagram of a controller circuit of the copier, channel select logic circuit of Fig. 3.

15 Fig. 7 is a block diagram of another switch.

Fig. 8 is a flow chart of a process for storing copies of a received data frame for the switches shown in Figs. 1 and 7.

20 Fig. 9 is a flow chart of a process for reading stored copies of a received data frame for the switches shown in Figs. 1 and 7.

Figs. 10a, 10b and 10c are representations of data stored in the switches of Figs. 1 and 7.

Like reference symbols in the various drawings indicate like elements.

## DETAILED DESCRIPTION

As shown in Fig. 1, a switch 100 has four ports 105a-105d. Ports 105a-105d are circuits may include hardware,  
5 software, firmware or any combination thereof. The ports 105a-105d are coupled to four buses 110a-110d respectively, and are used to transmit data from and receive data into switch 100. Ports 105a-105d and buses 110a-110d are full duplex in that they can transmit and receive data simultaneously. In  
10 one implementation, switch 100 is an Ethernet switch.

A receiving bus 115a and a transmitting bus 115c are coupled to ports 105a-105d. Receiving bus 115a forwards received data frames from ports 105a-105d to a copier, channel select logic circuit (CCSLC) 120. An intermediate bus 115b  
15 forwards received data frames from CCSLC 120 to main memory 125, and a bus 150 forwards address data to main memory 125 for use in storing the received data frames. The transmitting bus 115c forwards data frames stored in main memory 125 to ports 105a-105d. Interspersed in switch 100 are four output control  
20 logic circuits (OCLCs) 130a-130d that are associated with ports 105a-105d, respectively. CCSLC 120 is coupled to the four OCLCs 130a-130d and main memory 125 through control signal paths. It should be noted that a logic circuit may

include gates, hardware, software, firmware or any combination thereof.

Memory 125 includes channels and segments. A channel is a portion of the total width of a memory unit. A segment is an individually addressable collection of channels (e.g., a segment may include four channels). A location is a part of memory that is addressable by both a channel address and a segment address.

In general, the switch 100 receives data frames on buses 110a-110d at ports 105a-105d. The received data frames then are forwarded to CCSLC 120 using receiving bus 115a. CCSLC 120 determines how many copies of the received data frame are needed for broadcast or multicast operations, copies the received data frame, and determines particular locations in main memory 125 for storing the copies of the received data frame. OCLCs 130a-130d determine when to output the stored copies of the data frame over buses 110a-110d through ports 105a-105d based upon control data received from CCSLC 120.

As shown in Fig. 2, exemplary port 105a contains control circuit 210 and multiplexers 220. Exemplary port 105a receives a data frame on transmitting bus 115c for forwarding. The data frame received on transmitting bus 115c is extracted from transmitting bus 115c by multiplexers 220. Control

circuit 215 exchanges control signals with CCSLC 120 and OCLCs 130a-130d.

As shown in Fig. 3, CCSLC 120 includes memory 305 coupled to a controller 310. The controller 310 is coupled to a copier circuit 315 that receives incoming data frames on receiving bus 115a and outputs a number of copies of the incoming data frames onto intermediate bus 115b. The incoming data frames on receiving bus 115a contain header data and payload data. The output data frames on intermediate bus 115b are exact copies of the data frames received on receiving bus 115a and, accordingly, also include header data and payload data.

As described earlier, exemplary header data may include at least the destination address of the data frame and the length of the data frame. Controller 310 uses the destination address and length data fields to determine how many copies of the received data frame are required and where in main memory 125 to store those copied data frames. The controller 310 forwards addresses on address bus 150 to identify locations in main memory 125 to be used. In addition, the controller 310 generates control signals that are sent to OCLCs 130a-130d and used to determine which data frames to read out from main memory 125 and at what time to transmit those data frames.



Memory 305 is used to track locations in main memory 125 that are available to accept copies of received data frames. Memory 305 can be any suitable digital storage device. In one implementation, memory 305 is a volatile memory that can be  
5 accessed quickly for both reading and writing. The size of memory 305 correlates with the size of main memory 125.

As shown in Fig. 4a, an exemplary memory 305 may include an array of memory cells 405, a channel decoder 410 and a segment decoder 415. In one implementation, array 405 is four  
10 bits wide and sixteen bits long. This correlates to main memory 125, which has four channels and sixteen segments.

Each cell in array 405 holds a single bit and correlates to a particular channel in a particular segment of main memory 125. If a particular cell in memory 305 currently stores a 1-  
15 bit, that is an indication that the corresponding channel of the corresponding segment of main memory 125 contains valid frame data and cannot presently accept a new data frame. Alternatively, if a particular location in memory 305 currently stores a 0-bit, that is an indication that the  
20 corresponding channel of the corresponding segment of main memory 125 contains invalid frame data, (i.e., it is empty) and can presently accept new data.

Each cell in array 405 is individually addressable through channel decoder 410 and segment decoder 415, which

receive control signals from the controller 310. The cells also may be addressable by row or column in another implementation.

As shown in Fig. 4b, each cell 420 of array 405 may be implemented as an S-R flip flop that is enabled by a combination of the appropriate channel decoder and segment decoder signals. The set input of the flip-flop is connected to a write signal, and the reset input is connected to a read signal. Thus, the value of the cell is set to one when the write signal is asserted and the channel decoder and segment decoder signals indicate that data is being loaded into the corresponding portion of memory 125, and is set to zero when the read signal is asserted and the decoder signals indicate that data is being read from the corresponding portion of main memory 125. The cell 420 may be further configured to produce an output only when the enable signal is asserted so as to permit the controller to poll the memory 305 to detect cells having values indicative of available portions of main memory 125.

An exemplary main memory 125 may have four channels, each of which is 64 bytes wide, and sixteen segments. This means that main memory 125 can store 64, 64-byte data frames (one in each channel in each segment), sixteen, 256-byte data frames

(one spanning all four channels in each segment), or other combinations.

Fig. 5 shows a pair of exemplary cells 550, 555 in main memory 125 that each store 64 bytes. Each cell represents one location in main memory 125 (i.e., one channel of one segment). A decoder 560 uses the address bus 150 to generate signals that are combined with read and write signals to enable writing to and reading of a particular one of cells 550, 555. It should also be noted that any type of data randomly accessible, writeable storage device. Examples include RAM, SRAM, DRAM, RDRAM and SDRAM.

When a data frame is received, a determination is made as to which portion of main memory 125 is to store the received data frame. The received data frame then is copied onto bus 115b and the address bus 150 is used to identify one or more appropriate cells. The appropriate cells are activated by a combination of a signal from the decoder 560 and a write enable signal from the controller 310. Similarly, a stored data frame is forwarded onto bus 115c by signals on the address bus 150 identifying the appropriate cells. The cells are activated by a combination of a signal from the decoder 560 and a read enable signal from the controller 310.

As shown in Fig. 6, an exemplary controller 310 includes a size converter circuit 605, a channel availability circuit

610, a segment availability circuit 615, an arbiter circuit 620, and a frame address generator 625. Size converter circuit 605 receives a portion of the header data from a data frame on receiving bus 115a. Size converter 605 includes a  
5 size determiner circuit 605a, an address/port resolver circuit 605b, and an adjuster circuit 605c. Channel availability circuit 610 receives all of the bits stored in memory 305 as inputs. In the exemplary circuit shown in Fig. 6, memory 305 outputs 64 bits. Segment availability circuit 615 receives  
10 output data from size converter circuit 605 and channel availability circuit 610. Arbiter circuit 620 receives output data from size converter circuit 605, channel availability circuit 610 and segment availability circuit 615. Frame address generator 625 receives signals from OCLCs 130a-130d  
15 and outputs signals to main memory 125 and memory 305.

Size converter circuit 605 receives the size data of the data frame being transmitted on receiving bus 115a. Size data are extracted from the header data for a frame.

Since main memory 125 is divided into discrete channels,  
20 a determination is needed as to how many channels are required to store one copy of the received data frame. Using a maximum data frame width of 256 bytes, a standard binary encoding technique requires an 8-bit word to indicate the size of the data frame. Size converter circuit 605 takes the 8-bit word

for a particular frame and generates an output indicative of how many channels of a single segment in main memory 125 are required to store the frame. As an example, if the data frame on receiving bus 115a is 176 bytes wide, size converter

5 circuit 605 generates an output signal indicating that this data frame requires three channels (each of which is 64 bytes wide) of the segment.

Address-port resolver circuit 605b receives at least the destination address(es) from the header, and uses this data to  
10 produce an output indicative of which ports 105a-105d are to transmit the received data frame. The output of the address-port resolver circuit 605b is used to control operation of other parts of switch 100 so as to produce the correct number of copies of the received data frame and routing of the copies  
15 of the received data frame to the proper port or ports.

Adjuster circuit 605c receives the output from size determiner circuit 605a and the output from address-port resolver circuit 605b and generates an output indicative of the number of segments needed to store the number of copies of  
20 the data frame. For example, if a 128-byte received data frame is to be broadcast through all four ports, such that four copies of the received data frame are to be stored in main memory 125, the adjusted 605c will indicate that the data

frame is to be stored in at least two segments, with each copy being stored in two channels of a segment.

Channel availability circuit 610 receives the outputs from memory 305 and determines where there are vacant  
5 channels. This is accomplished using logic gates or the like to determine sets of zeroes in a segment of memory 305. That is, channel availability circuit 610 determines if there are any single zeroes, pairs of zeroes, triples of zeroes or quads of zeroes. The results are output to segment availability  
10 circuit 615 and arbitrator circuit 620. Depending on the design of channel availability circuit 610, it may only register availability if contiguous channels in a segment are vacant or it may register availability if the proper number of channels is available regardless of contiguousness.

15 Arbitrator 620 selects the channel(s) and segment(s) to receive the copies of the input data frame. Typically, main memory 125 will have multiple locations that are available to receive the copies of the input data frame. As an example, if switch 100 receives a 64-byte data frame for which only one  
20 copy is needed, and main memory 125 is empty, there are 64 locations that can accept this data frame. Arbitrator 620 selects one location from the available pool of 64 locations.

Each OCLC stores data that are in turn used to determine when each port is to receive and output a stored data frame.

In one implementation, each OCLC receives a code word that translates to one or more locations in main memory 125. This code word is stored in a queue. As the data are output from main memory 125 to the port associated with the OCLC, the code words stored in the queue are advanced. When a particular code word in the OCLC reaches the head of the queue, it is forwarded to frame address generator circuit 625.

In the implementation shown having four ports, it is possible that two or more OCLCs will request data from the same channel or segment at one time. To avoid this collision of data from main memory 125, the frame address generator circuit 625 arbitrates between the received code words.

Fig. 7 shows an alternative switch 700 that includes a general processor 720. Like exemplary switch 100, switch 700 includes four ports 105a-105d that are coupled with four external buses 110a-110d and internal buses 115a-115c. Processor 720 is coupled with internal bus 115a and intermediate bus 115b. Memory 125 is coupled with internal buses 115b 115c.

The function and operation of most of the elements of exemplary switch 700 have been previously described and will not be repeated. One difference between exemplary switches 100 and 700 is the use of a general purpose processor to perform the copying, the determining of acceptable memory

locations to store the copies, and the outputting of data frames from memory 125 to ports 105a-105d for transmission over buses 110a-110d. Processor 720 contains memory such as ROM or RAM (not shown) that holds the instructions used to control the operation of processor 720 and therefore the operation of switch 700.

Fig. 8 shows an exemplary process 800 for copying data frames into a memory. This process is initiated when the switch receives a data frame (step 805). The header information, which contains at least destination information and frame size, is extracted from the data frame (step 810). From the destination information, the number of ports that are to output the data frame is determined (step 815) along with the size of the received data frame (step 820). From the number of ports to output the data frame and the size of the received data frame, the minimum amount of memory necessary to store the copies of the data frame, with one copy for each port that is to output the data frame, is determined (step 825).

Next, the empty locations in memory are determined (step 830). In one implementation, the determination is made by storing 1-bits and 0-bits in a separate memory, with the 1-bits and 0-bits corresponding to full and empty locations, respectively, in the data frame memory, and polling this



separate memory to locate an adequate concentration of 0-bits  
empty locations to store the copies of the received data  
frame. Once all of the suitable locations in frame memory  
have been identified, one or more locations are selected to  
5 store the copies of the data frame (step 835). The copies of  
the data frame are then stored in the selected memory  
locations of the frame memory (step 840). Each copy of the  
data frame is associated with a port that will transmit that  
copy of the data frame and this association is stored along  
10 with the locations in frame memory of the copies of the data  
frame (step 845). The process then ends (step 845).

Fig. 9 shows an exemplary process 900 for outputting data  
frames from a switch. The process begins when multiple  
associations are selected (step 905). In other words, every  
15 port of the switch is polled to determine if they have a data  
frame in frame memory that is ready to be transmitted. One  
exemplary method of performing this step is to store the  
associations in a queue and read them in a first-in-first-out  
(FIFO) order.

20 With multiple ports requesting data from the frame memory  
at the same time, it is possible that a conflict may arise  
such that two ports will require data from locations that  
share a data output bus in the frame memory. Accordingly, a  
determination is made to see if there is a conflict (step

910). If there is no conflict such that every port that has data frames to output may do so simultaneously, then the data frames are read from the frame memory in parallel (step 915). The ports select the data frames that are to be output, based  
5 on the association described above, and output the selected data frames (step 920). The process then ends (step 925).

If a conflict is determined (see step 910), then one of the ports that has a conflict is instructed to wait (step 930) and the remaining associations are checked again for a  
10 conflict (step 910). At worst case, ports will be instructed to wait until only one port remains and then the remaining port will be able to retrieve and output its data frames freely (see steps 915-925).

As shown in Fig. 10a, four data frames, each only a  
15 single channel wide, are stored in a segment 1005 of main memory in a conventional switch. The data frames are forwarded to copier 1010 where the necessary copies are made. In this example, two copies of each data frame are generated before the data frames are output to the ports. Since copier  
20 1010 can only copy one data frame at a time, it follows that four read cycles are needed to output the four data frames L-O from Segment 9. It should be noted that if copier circuit 1010 were not present, the number of read cycles to output the four data frames shown increases to eight. It should also be

noted that if multiple data frames are not packed into a single segment, the use of the memory would be less efficient.

In contrast, Fig. 10b, shows two segments, 1015 and 1020, which have multiple copies of data frames already in them, in accordance with the implementations described above. Due to the packing of all of the needed copies of the data frames into a single segment, the implementations described above will output the required L and M data frames in one read cycle and the N and O frames in another cycle. Assuming that there are no port collisions such as the L data frame and the M data frame being transmitted over the same port, the implementations described can transmit the needed data frames in half the time of conventional switches (i.e., in two read cycles).

In one implementation, multiple segments of main memory may be accessed simultaneously. This also provides increased efficiency. In Fig. 10c, the K and L data frames are read and transmitted first and the M data frames are read and transmitted second. Suppose that the arbitration circuit or process places the N data frames and the O data frames in segments 4 and 5, respectively. If multiple segment access was not allowed, two read cycles would be needed to output the N data frames and the O data frame. However, in the implementation described above, multiple segment access is

permitted and the N and O data frames can be read simultaneously if they do not have a port conflict.

Another advantage of the described implementations over conventional switches is the ability to determine where there is adequate memory space in a segment to accept the data frame. In some implementations, the described circuit and process are systematic like a queue. In other words, the data frames are stored in main memory 125 as they are received and little packing of different frames into a single segment is done (i.e., each received data frame and its corresponding copies are stored in a single segment). However, in other implementations, the circuitry looks for available space and only arbitrarily selects channel(s) or segment(s) when a conflict arises. Thus, the data frames are not stored in a strict queue-like structure in main memory 125. Instead, the described implementations can pack different data frames or multiple copies of a data frame (one per channel) into a single segment to allow for increased bandwidth when forwarding data frames over different ports in one read cycle.

It should be noted that other details are implicit in the above descriptions. For instance, multiple ports are receiving data frames and attempting to have them stored in memory. In the exemplary circuit described above, two or more ports 105a-105d may attempt to simultaneously place a data

frame on receiving bus 115a. This internal data collision can be avoided by implementing well-known techniques.

In addition, most switches have error detection/correction. One example of an error is receiving  
5 only part of a data frame. Most switches, including those that implement circuits and techniques similar to those described above, will have error detection/correction techniques that will discard a received erroneous data frame and prevent it from being stored and/or transmitted in the  
10 network.

Similarly, common error correction methods in broadcast and multicast situations may also be employed. For example, broadcast storms, where data frames are replicated and sent back and forth between two switches until there is no room for  
15 new data frames, can be avoided by using protocol techniques that will cause the excess data frames to be considered as error data frames and not copied nor forwarded.

It should also be noted that the above discussion did not describe the use of a clock. Some implementations will  
20 include a clock to help manage data flow and avoid collisions. Some implementations will use clocks and registers to make the above calculations in pipeline format.

It will be understood that various modifications may be made. For example, main memory 125 need not be centralized in

switch 100, but instead the main memory can be distributed across switch 100. In this alternative implementation, the copies of the received data frame are first routed to the distributed memory associated with the ports from which the  
5 copies are to be output. In yet other memory structures used in other implementations, input buffers and/or output buffers are added to ports 105a-105d to assist in the management of the data frames.

Other implementations include half-duplex buses 110a-110c  
10 instead of the full duplex buses described above. These implementations require collision avoidance, which can be obtained using the set of rules called carrier sense multiple access with collision detection (CSMA/CD).

Other implementations may include different numbers of  
15 ports, channels and segments in the memories. Nor do the number of channels in the memories need to equal the number of ports in the switch. Similarly, alternative implementations could require that each port be assigned to one channel in the memories.

20 In yet another implementation, copies of data frames that are more than half as wide as memory 125 are written into the segments in multiple write cycles instead of the single write cycle described above.

Other implementations may use fixed-length data frames. In such implementations, size determiner 605a is replaced with a constant value output that is the frame size of the implementation.

5        The above systems and methods may be implemented in any of a number of networking protocols. Some of these protocols may modify the header data, the payload data or both. Accordingly, other implementations are within the scope of the following claims.

10